

DOCUMENT RESUME

ED 263 906

IR 011 895

AUTHOR Rand, David; Kimball, Richard L.
TITLE A Programming Contest for Grades 4-12.
PUB DATE 84
NOTE 22p.
PUB TYPE Guides - Non-Classroom Use (055) -- Reports - Descriptive (141)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Competition; Computer Literacy; *Computer Software; Elementary Secondary Education; Microcomputers; *Problem Sets; *Programing; Programing Languages; Success

IDENTIFIERS BASIC Programing Language; *Contests; *Debugging (Computers)

ABSTRACT

In April of 1984 the Aroostook (Maine) Computer Educators, in conjunction with the University of Maine at Presque Isle, conducted a computer programming contest for county students in grades 4-12. This document describes the structure of that contest, including grade level groupings (4-6, 7-9, and 10-12); rules for team membership; the schedule for the contest; and the three major events: the programming challenges, debugging contest, and the individual entries. Comments on the contest are also included. The major part of the document consists of the appendices, which list the three sets of programming challenges used in the contest by grade level. (JB)

* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *

A PROGRAMMING CONTEST FOR GRADES 4-12

Richard L. Kimball, Coordinator
Academic Computing Services
University of Maine at Presque Isle

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

* This document has been reproduced as
received from the person or organization
originating it.

□ Minor changes have been made to improve
reproduction quality.

• Points of view or opinions stated in this docu-
ment do not necessarily represent official NIE
position or policy.

David Rand, Principal
Mapleton Elementary School
Maine School Administrative District #

"PERMISSION TO REPRODUCE THIS
MATERIAL HAS BEEN GRANTED BY

David P. Rand

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)."

ABSTRACT

A programming contest held last spring is described including the rules sent to prospective entrants. The structure is reported along with some comments. The actual contest problems are included as appendices.

In April of 1984 The Aroostook Computer Educators in conjunction with the University of Maine at Presque Isle, conducted the first annual programming contest for Aroostook County students. We found this event to be both an intellectual stimulus and an exciting 'sporting' challenge. Many children throughout Aroostook County are already thinking about and planning for the spring 1985 contest. Several districts that did not participate last year are planning to become involved this year.

DESCRIPTION OF THE CONTEST

The contest had three major events. These were the individual submissions, programming challenges, and debugging. The contestants were broken into three categories by grade level.

In the grade 4 - 6 category three programming teams and four debugging teams came from four separate school districts, and there were three individual program submissions. In grades 7 - 9, each of five schools entered a programming and a debugging team. For all but one school the two teams had the same membership. Twelve individual programs were entered. At the grade 10 - 12 level seven schools entered both programming and debugging teams with a great deal of overlap between each school's two teams. Also four individual programs were submitted. In all, over seventy students participated actively in the contest as well as a few individuals who acted as alternates for the team members. Each district sent coaches for their teams and its own equipment. We were quite liberal with awards in order to encourage participation and give satisfaction to the entrants. Each team event had a winning team and a runner up at each level and in the individual entries there

ED263906

IR011895

awards to approximately half the entrants. We gauged the number of awards by the number of entrants at each level.

We include below the contest rules as they were sent to the participating school districts.

Grade level groupings are (1) grades four to six, (2) grades seven to nine, and (3) grades ten to twelve. If your school set up makes it more convenient, you may include a student from a lower grade level, but not from a higher grade level.

Each school may send one team each per grade level grouping for the programming challenges and for the debugging contest or the same team may do both contests. A team consists of three members. If your school has more than one of these grade level groupings, separate teams may be sent for each grouping. We recommend that alternates also be prepared in case of last minutes illness. If you have a good number of interested students a good preparation would be to have a school contest.

Each school should bring their own computer(s). Each team will be allowed one computer set-up. Hardware only should be included---no manuals, books, etc. about programming should be present. Bring an extension cord with multiple plug-ins at the end! Hardware may include the computer, TV or monitor, cassette or disk storage. Do not bring a printer.

PROGRAMMING CHALLENGES

Grade groupings will be placed together in rooms, with judges only (supplied by UMPI). Each team will receive the same list of programming challenges, from which they may select any they wish and work on them one at a time. As they complete a problem, they will signal the judge who will check the program for meeting the output requirements. If the judge agrees, they proceed to another challenge. The point will be to complete as many challenges as possible in the allotted time. Besides the winning team, there may be second, and third place prizes as well, depending on the number of teams entered.

DEBUGGING CONTEST

Each team will be given the same program, which once was a working program that has had a

number of bugs placed into it. The program will be on a print-out and must first be entered into the team's computer. The first team in each grade grouping to get the program working properly is the winner for the grouping. Again, there will be second and third place prizes if several teams enter.

INDIVIDUAL ENTRIES

Students may submit one program each. Programs may include graphics but should not just be graphics. Programs must be mailed or sent to with only the name of the micro-computer required and the grade grouping written on the cassette or disk. The container for the cassette or disk should include school, grade grouping, and sponsoring teacher. Also included should be a printout of the program. If a printer is not available, a handwritten copy will be acceptable. Finally, include the individual program entry form included with this bulletin. [Note: The entry form included a place for the instructor to certify that the program was original and developed by the student.] The above information should not be in the program such as in the title or in REM statements. The programs will be coded so that the judges cannot know the school of the student, thus ensuring objectivity. The deadline for individual entries will be (date three weeks prior to team events). If the computer used is not easily available to the judges at UMPI the teacher may be requested to supply one for a week or so. Schools should be selective about programs entered. In other words, we would not expect to see everyone in a class submit one as part of an assignment. Teachers should do some screening.

The individual entries will be judged prior to the Programming Contest date. The individual winner announcements will be made after lunch along with the winners of the other contests. Of course, names will be selected in each grade grouping, as well as second and third prizes, depending on the number of entries. It is planned that the winning programs will be demonstrated.

The schedule on the day of the contest will be as follows.

9:00 - 9:30 Registration and setting up of equipment.

9:30 - 9:45 Opening remarks and instructions to contestants.

10:00 - 11:00 Programming challenges.

11:15 - 12:15 Debugging contest.

12:15 - 1:00 Lunch (provided by the University)

1:00 - 1:30 Awards to team or individual contestants.

After the awards, time will be provided for the winning individuals to demonstrate their program entries to interested parties. Also campus tours will be available for anyone wishing to participate.

COMMENTS ON THE CONTEST

Some of the difficulties in providing appropriate contest materials were as follows. The students from the various schools have differing backgrounds and experience with various machines and versions of BASIC. Although no stipulation was made that BASIC would be the language used (except in the debugging contest), it was used exclusively. Perhaps some year a team will come prepared to use another language and can show its superiority by improved performance in the contest.

Many of the problems have a mathematical bent. This is unfortunate but very difficult to avoid. Most versions of BASIC agree on the mathematical facilities but differ greatly on graphics, strings, random number generation, and file handling. Unfortunately for the contest, these are exactly the parts of programming to which students are often first exposed. Another year we hope to alleviate some of these problems by having suggestions for problems submitted in advance by the participating schools. This will help to broaden the scope of the problems and to increase the 'investment' the schools have in the contest.

Each team was required to bring and set up its own equipment. All that we provided were tables and a few extension cords. The only difficulty this seemed to cause was that one team forgot to bring its extended BASIC module and had to participate with the weaker version built into their machine.

Judges for the event were selected from among University students who had experience programming in BASIC and at least one other language. They were supervised by the University coordinators of the event. Four or five teams at each grade level were placed in one room with two judges.

If there had been more teams more judges would have been necessary.

In the programming challenges part there were instances in which more than one team solved the exact same number of problems. In these cases we broke the tie by subjectively weighting the problems solved for their difficulty. It would perhaps have been advantageous to have worked this out in advance. It would then be possible to let the teams know what the weighting was so they could plan their strategies accordingly.

In the debugging contest we had originally planned to have the bugged programs available to the teams either on disk or tape to avoid the program entering process. This proved infeasible due to the variety and unavailability to the contest officials of the machines actually used. By presenting the bugged programs in hardcopy form the contest then included speed and accuracy of program entry. This seemed to create little problem, in fact most teams combined program entry with the initial stages of debugging. Incidentally, debugging seemed to divide the teams widely with one or two teams finishing very rapidly and the others either unable to finish or taking most of the time available.

Individually submitted programs were required to be submitted about three weeks prior to the team part of the competition. These were submitted on disc or tape accompanied by a printed listing through a third party in order to provide a buffer of anonymity between the judges and the contestants. This was not altogether perfect since some of the programs had been prepared in part on the University computers and were recognized by some of the judges. Several sessions were necessary in order to determine the winners in the individual entry portion largely due to the difficulty in getting the taped submissions to load. In many cases we had to trace back to the submitter and have the actual tape recorder used in making the tape delivered to us in order to view the program in operation. One factor which would probably help this situation would be to request that if at all possible the programs be submitted on disk. We were also involved with a lot of borrowed equipment as the University did not own the actual computers needed. This problem has been remedied to some extent as we have since purchased some of the low end computers common in the local area schools.

DESCRIPTION OF APPENDICES

Attached you will find the three sets of programming challenges used in the contest. Each has approximately fifteen problems. This number seemed to be sufficient for the sixty minutes allowed. The problems come from several

sources, but the excellent book, BASIC/APPLE 11e, A PROGRAMMING GUIDE, BY Allen B. TUCKER, Jr. should be mentioned. The three programs with the built in bugs are also attached. The coin counter program was used for the grade 4 - 6 level and was probably too difficult. Only one team was able to debug it and it took them a long agonizing time. The Anglo to metric program was for the middle level and was completed by the first team in a very short time. The 9 - 12 level problem involved prime numbers with which some of the teams seemed to be unfamiliar.

Programming Challenges
Grades 4-6

1. Write a BASIC program to READ three numbers from a DATA statement and then print out the smallest.
2. Write a program to count and print out by fives beginning with 0. The first few numbers will be 0, 5, 10, 15 and so on. Interrupt the program manually when forty or fifty numbers have been printed out.
3. Write a program to read variables A, B, C, and D from numbers of your choice in a DATA statement. Print out the numbers vertically with B below A, C below B, and D below C.
4. Write a program that will read the four numbers 10, 9, 1 and 2 from a DATA statement, putting the numbers in A, B, C, and D respectively. Add the first two numbers putting the sum in S. Then compute the product of the last two numbers, putting the result in P. Print out the value of S and P on the same line.
5. PROBLEM - Reverse Order
Read four numbers and display them in reverse order.

INPUT

The four numbers

Sample input:

3
5
2
104

OUTPUT

The numbers in reverse order.

Sample output:

NUMBERS IN REVERSE ORDER

104
2
5
3

BEST COPY AVAILABLE

6. Write a program to print out the following pattern of asterisks:

```
*****
*****
***
*
```

No fair using more than one PRINT statement!

7. Write a program to accept the input of a number N, then print out the even numbers greater than 0 but less than or equal to N.
8. Write a program to generate a table of numbers and their square roots. The table should look as follows:

N	SQR(N)
2.0	1.41421
2.1	1.44914
2.2	1.48324
etc.	
3.9	1.97484
4.0	2.00000

9. Write a program that will print out the number pattern shown below and then stop.

0	5	10	15	20
25	30	35	40	45
etc.				
150	155	160	165	170

10. Write a program to compute and print out the sum of all the whole numbers between 1 and 100 inclusive.
11. Write a program which will accept as data a list of numbers of any length and determine the largest of these numbers.
12. Suppose you wanted to sort Celsius temperatures into pleasant and unpleasant temperatures. Let's say the pleasant range is between 15 to 25. Write a program that will take a number as data and then will print out "ITS NICE" with the number if it falls in the pleasant range but will print out "STAY HOME" with the number if it does not.

BEST COPY AVAILABLE

13. Write a program to count and print out by tens going from 0 to 500.
14. Write a program to accept the input of two numbers. If both the numbers are greater than or equal to 10, print out their sum. If both the numbers are less than 10, print out their product. If one number is greater than or equal to 10 and the other is less than 10, print out the difference between the largest and smallest.

Programming Challenges
Grades 7-9

1. Write a program to simulate 1000 throws of a die and tabulate the results showing how many falls were obtained for each number from 1 to 6.
2. Write a program to determine by division whether a given number is prime. The number is input and the result is either that the number is prime or its smallest factor is printed.
3. Write a program to input a list of 20 numbers and print out the number of:
positive numbers
negative numbers
zeros
numbers greater than 50
number of even numbers
numbers between -10 and +10
4. PROBLEM - Compute Gross Pay
Write a program that reads the number of hours worked and the salary rate per hour of an employee and computes that employee's pay.

INPUT

Hours worked and salary rate per hour.

Sample input:

23.0

4.50

OUTPUT

The resulting gross pay.

Sample output:

Gross=103.5

5. PROBLEM - Exam Score Average
Write a program that reads three exam scores and finds their average.

INPUT

Three exam scores

Sample input:

97.5 75.3 89.0

OUTPUT

Their average

Sample output:

AVERAGE = 87.2667

BEST COPY AVAILABLE

6. PROBLEM - Cost of a Trip

Given the distance traveled, the miles per gallon that the car gets, and the cost per gallon of gasoline, calculate the cost of the gasoline used for the trip.

INPUT

The distance traveled, miles per gallon for the car, and the cost per gallon of gas.
Sample input:

185.5 28.4 1.319

OUTPUT

The cost of the trip.

Sample output:

COST OF THE TRIP = 8.6153

7. An instructor decides to award letter grades on an examination as follows:

90-100	A
80-89	B
60-79	C
50-59	D
0-40	F

Write a program to produce the following output when executed:

INPUT EXAM GRADE? (You type in numerical grade)

YOUR GRADE IS (Computer types out A, B, C, D, or F)

8. If you get a ticket for speeding, your fine is based on how much you exceeded the speed limit. Suppose the fine is computed as follows:

<u>Amount over Limit</u>	<u>Fine</u>
1-10 mi/h	\$ 5
11-20	10
21-30	20
31-40	40
41 or more	80

Write a BASIC program that will produce the following output when executed.

WHAT WAS SPEED LIMIT? (You type in)

SPEED ARRESTED AT? (You type in)

FINE IS (Computer types out fine) DOLLARS

9. Write a BASIC program to call for 12 grades to be input. Compute and print out (1) the highest grade, (2) the lowest grade, (3) the average of the grades.

BEST COPY AVAILABLE

10. Write a program to find the average of all the positive numbers in a list whose end is marked with the flag 9999. The list of numbers is to be typed in at the terminal at run time.
11. Write a program to fill a 3 by 4 array with ones. Then have the array printed out.
12. Write a program to generate and print out fifty integers selected at random from the range 13 to 25.
13. Write a program that will simulate tossing a coin, 10, 50, and 500 times. In each case, print out the total number of heads and tails that occur.
14. Construct a dice-throwing simulation in BASIC. Two dice are to be thrown twenty times. For each toss, print out the dice faces that are uppermost.
15. Write a program to generate and print out the average of 100 random numbers selected from the range 0 to 1.

BEST COPY AVAILABLE

Programming Challenges
Grades 10-12

1. Write a program which will read in twenty test scores and:

- (a) compute the mean (arithmetic average)
- (b) determine the highest and the lowest scores

For scores use: 75, 97, 58, 62, 88, 45, 82, 86, 85, 89, 70, 71, 72, 64, 67, 69, 78, 76, 74, 79

2. Suppose a man is given a job for which the salary is \$.01 the first day, \$.02 the second, \$.04 the third, \$.08 the fourth, etc. Suppose also that he starts with no money, has no expenses and saves all that he earns. Write a program to determine on which day he becomes a millionaire (and is consequently fired).

3. Write a program which will compute values of y to fill in the following chart for the equation $y = x^3 - 3x^2 + 2$

x	-1	-.5	-.75	-.25	0	.25	.5	.75	1	1.25
y										

4. Suppose a piece of paper .005" thick could be folded any number of times. How many times must it be folded to make a thickness sufficient to reach the moon assuming the moon is 200,000 miles from earth? Write a program to solve this problem.
5. Write a program to print out and count the primes from 1 up to 1000.
6. Write a program to input a , b and c and solve the quadratic equation $ax^2 + bx + c = 0$ for any real roots
- If the equation does not have real roots an appropriate message should be printed.
7. Write a program which will accept a list of 10 integers as input and will print them out in ascending order.

BEST COPY AVAILABLE

8. PROBLEM Bar Charts

Write a program that reads a series of numbers which represent the scores achieved by a class of students on a particular exam and produces a bar chart of the results.

INPUT

A series of numbers (between 0 and 100) that represent the exam scores for a class of students.

Sample Input:

86.5
94.2
37.5
100.0
89.0
89.2
93.1
72.4

OUTPUT

A bar graph of the scores. Graphics should be used to draw the lines.

Sample Output:

86.5-----
94.2-----
37.5-----
100.0-----
89.0-----
89.2-----
93.1-----
72.4-----

9. PROBLEM - Inner Product

An inner product (or dot product) is the sum of the products of the corresponding elements of two one-dimensional arrays. For instance, the inner product of two 3-element arrays is

$$A(1)*B(1) + A(2)*B(2) + A(3)*B(3)$$

Write a program that computes the inner product of two arrays of any size.

INPUT

The values for the two arrays.

Sample Input:

26.50	47.314
-421.75	-61.04
27.00	-1.050
13.40	102.50
-51.70	21.004
16.41	21.51
13.00	13.00

OUTPUT

The size of the arrays to be multiplied and the inner product.

Sample Output:

THE ARRAY SIZE IS: 7

THE INNER PRODUCT IS: 27778.6

NOTE

Assume that the maximum size of each array is 200 elements.

BEST COPY AVAILABLE

10. PROBLEM - Factorials

The factorial of a positive integer (represented as $N!$) is the product of all positive integers from 1 to N .

$$N! = N(N - 1)(N - 2) \dots 1$$

Hence the factorial of 6 is

$$6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$$

and the factorial of 7 is

$$7! = 7 \cdot 6! = 7 \cdot 720 = 5040.$$

Write a program that reads a series of numbers and computes the factorial of each.

INPUT

An unknown number of positive integers, followed by the trip value 9999.

Sample input:

7

6

14

3

1

9999

OUTPUT

The corresponding factorials.

Sample output:

7 FACTORIAL = 5040

6 FACTORIAL = 720

14 FACTORIAL = 8.71782E10

3 FACTORIAL = 6

1 FACTORIAL = 1

11. PROBLEM - Convert Binary to Decimal

Write a program that reads a series of numbers in base 2 (binary) and converts them to base 10 (decimal).

INPUT

A series of 8-digit binary numbers.

Sample input:

10101111

11111111

00001010

OUTPUT

The equivalent decimal values.

Sample output:

BASE 10 = 175

BASE 10 = 255

BASE 10 = 10

BEST COPY AVAILABLE

12. PROBLEM - Array Search

Given an array of numbers, find the average of the numbers and then find the number in the array that is closest to the average.

INPUT

The values for the array (assume that the maximum number of values in the array is 100).

Sample input:

86.0
84.0
87.0
92.0
100.0
71.0
77.0
96.0

OUTPUT

The actual array size, the average, and the value of the element of the array closest to the average.

Sample output:

THE ARRAY SIZE IS: 8
AVERAGE = 86.625
CLOSEST VALUE = 87.0

NOTE

To search an array ARR for the element closest to the average, use a variable, say CLOSE, to hold the closest value found so far. Set CLOSE=ARR(1) as an initial assumption. Next go through the rest of the array comparing CLOSE with ABS(ARR(I)-AVERAGE) for each I.

13. PROBLEM - Fibonacci Sequence

There is an interesting sequence of numbers called the Fibonacci numbers. The set begins with 0, 1. Then each succeeding number in the sequence is the sum of the two previous ones. Thus, the Fibonacci sequence is

0, 1, 2, 2, 3, 5, 8, ...

Write a BASIC program to compute and print out the first twenty numbers in the Fibonacci sequence.

14. PROBLEM - Area of a Triangle

Read in the lengths of the three sides of a triangle and compute its area. The equation to find the area of a triangle is:

$$\text{Area} = \sqrt{S(S-A)(S-B)(S-C)}$$

where A, B, and C are the lengths of the sides and S is the semiperimeter of the triangle

$$S = \frac{A+B+C}{2}$$

BEST COPY AVAILABLE

14. (continued)

INPUT

A, B, and C, the lengths of the sides of a triangle.

Sample input:

6.0 8.0 4.0

OUTPUT

The area of the triangle.

Sample output:

AREA = 11.619

NOTE

In order to have a triangle, no one side may be longer than the sum of the other two sides. Your program should test for this possibility. If the given lengths do not define a triangle, then display zero for the area.

15. PROBLEM - Integer Remainder

A very useful subroutine is one that computes the integer remainder, or modulus, when dividing one integer (PA) by another (PB). This is often found as a built-in function in programming languages, but it is not part of every BASIC. Write a subroutine that will accomplish this, and then write a main program that will exercise the subroutine.

INPUT

A series of values of A and B.

Sample input:

13 5
22 9
-50 3
12 6

OUTPUT

The corresponding modulus for each pair of values.

Sample output:

A MOD B = 3
A MOD B = 4
A MOD B = 2
A MOD B = 0

NOTES

1. The modulus should always be computed as a positive number, regardless of whether the two numbers in question are both positive or not.
2. Recall that $\text{INT}(\text{PA}/\text{PB}) * \text{PB} = \text{PA}$ only when PA is an exact multiple of PB. Otherwise, the difference between the two sides of this equation is identically the modulus of PA and PB.

BEST COPY AVAILABLE

16. PROBLEM - Checking Account Transactions

Write a program which, given a bank customer's ID number and the balance in his or her checking account, processes a series of transactions (deposits, withdrawals) to his or her account, displaying the new balance after each transaction.

INPUT

The customer number and the current balance in his or her account, followed by an unknown number of transactions (deposits are positive, withdrawals are negative), terminated by the trip value 9999.

Sample input:

```
146250      412.65
-104.00
-39.37
+315.85
-200.00
-55.00
9999
```

OUTPUT

The new balance resulting from each transaction as it is entered.

Sample output:

```
NEW BALANCE = 308.65
NEW BALANCE = 269.28
NEW BALANCE = 585.13
NEW BALANCE = 385.13
NEW BALANCE = 330.13
```

JLOAD BUGGY COIN COUNTER
JLIST

```
1  REM  MONEY COUNTING PROGRAM
2  REM  THIS CALLS A SUBPROGRAM FOUR TIMES TO FIND OUT HOW MANY OF EACH TYPE OF
   COIN YOU HAVE
3  REM  IT COMPUTES THE TOTAL VALUE FOR EACH KIND OF COIN AND THEN PRINTS OUT
4  REM  THE TOTAL DOLLAR VALUE OF THE COINS.
10 A$ = "PENNIES":V = 1
20 GOSUB 200
30 A$ = "NICKELS":V = 5
40  GOSUB 200
50 B$ = "DIMES":V = 10
60  GOSUB 200
70 A$ = "QUARTERS":V = 50
80  GOSUB 200
90 WRITE"THE TOTAL VALUE IS  $";T / 10
100 END
200 PRINT "NUMBER OF ";A$;"? ";
205 INPUT N
210 PRINT N;" ";A$;" = ";N * V;" CENTS"
220 T = T + N * V
230 GOBACK
```

YOU NEEDN'T TYPE IN THE REMARKS IF YOU DON'T WANT TO

JLOAD BUGGY ANGLO TO METRIC
JLIST

YOU NEEDN'T TYPE IN THE REMARKS IF YOU
DON'T WANT TO

```
1  REM  THIS PROGRAM CONVERTS A MEASURE GIVEN IN ANGLO UNITS TO METRIC UNITS.
HE  CONVERSIONS AVAILABLE IN THIS PROGRAM ARE AS FOLLOWS:
2  REM      1 INCHES TO CENTIMETERS
3  REM      2 FEET TO CENTIMETERS
4  REM      3 FEET TO METERS
5  REM      4 YARDS TO METERS
6  REM      5 MILES TO KILOMETERS
7  REM      6 DEGREES FAHRENHEIT TO DEGREES CELCIUS
10 PRINT "ANGLO TO METRIC"
20 PRINT
29 REM  - ESTABLISH VARIABLES FOR 6 CONVERSION FACTORS
30 DIM C(6)
39 REM  - LOOP TO ASSIGN CONVERSION FACTORS INTO C( )
40 FOR N = 1 TO 5
50 READ C(N)
60 NEXT N
69 REM  - DATA TABLE OF FIVE CONVERSION FACTORS
70 DATA 2.540, 30.480, .3048, .9144, 1.609
89 REM  - GET NUMBER OF CONVERSION FROM PROGRAM DESCRIPTION
90 PRINT "(TO END PROGRAM ENTER 0)"
100 PRINT "WHICH CONVERSION DO YOU NEED?"
110 INPUT N
119 REM  - END PROGRAM?
120 IF N = - 1 THEN 540
129 REM  - CONVERSION AVAILABLE?
130 IF N > 4 OR N < 0 THEN 100
140 PRINT "VALUE TO BE CONVERTED ";
150 INPUT I$
159 REM  - PERFORM CONVERSION USING PROPER CONVERSION FACTOR
160 R = I * C(N)
169 REM  - DIRECT PROGRAM TO PROPER CONVERSION UNITS. PRINT RESULTS
170 ON N GOTO 180, 200, 220, 240, 260, 500
180 PRINT I; " INCHES = "; R; " CENTIMETERS"
190 GOTO 520
200 PRINT I; " FEET = "; R; " CENTIMETERS"
220 PRINT I; " FEET = "; R; " METERS"
240 PRINT I; " YARDS = "; R; " METERS"
260 PRINT I; " MILES = "; R; " KILOMETERS"
499 REM  - CONVERT FROM DEGREES FAHRENHEIT TO CELCIUS
500 R = (I - 32) * 5 / 9
510 PRINT I; " DEGREES FAHRENHEIT = "; R; " CELCIUS"
520 PRINT
529 REM  - RESTART PROGRAM
530 GOTO 100
540 END
```

1

BEST COPY AVAILABLE

JLOAD BUGGY PRIME FACTORS OF INTEGER
JLIST

```
10 PRINT "PRIME FACTORS OF INTEGERS"
20 PRINT
30 PRINT "(ENTER 0 TO END PROGRAM)"
40 PRINT "NUMBER";
50 INPUT Z
59 REM - END PROGRAM?
60 IF Z = - 1 THEN 200
69 REM - THE SIGN OF THE NUMBER IS ALWAYS A FACTOR
70 PRINT SGN (Z)
79 REM - USE ABSOLUTE VALUE FOR CALCULATIONS
80 Z = ABS (Z)
89 REM - LOOP TO TEST ALL INTEGERS (2 THROUGH Z) AS PRIME FACTORS
90 FOR I = 2 TO Z
100 S = 1
110 IF Z / I = INT (Z / I) THEN 150
120 Z = Z / I
130 S = S + 1
140 GOTO 110
149 REM - FIND A PRIME FACTOR? IF YES, PRINT
150 IF S = 0 THEN 170
159 REM - PRINT FACTORS WITH EXPONENTS: I^S = I TO THE S POWER
160 PRINT I;"^";S
170 IF Z = 1 THEN 180
175 NEXT I
180 PRINT
189 REM - RESTART PROGRAM
190 GOTO 40
200 END
```

YOU NEEDN'T TYPE IN THE REMARKS IF YOU DON'T WANT TO.
THIS PROGRAM IS SUPPOSED TO FIND THE PRIME FACTORS OF ANY INTEGER Z WHICH IS INPUT.
THE RESULTS ARE PRINTED OUT SHOWING EACH FACTOR AND ITS EXPONENT.

BEST COPY AVAILABLE